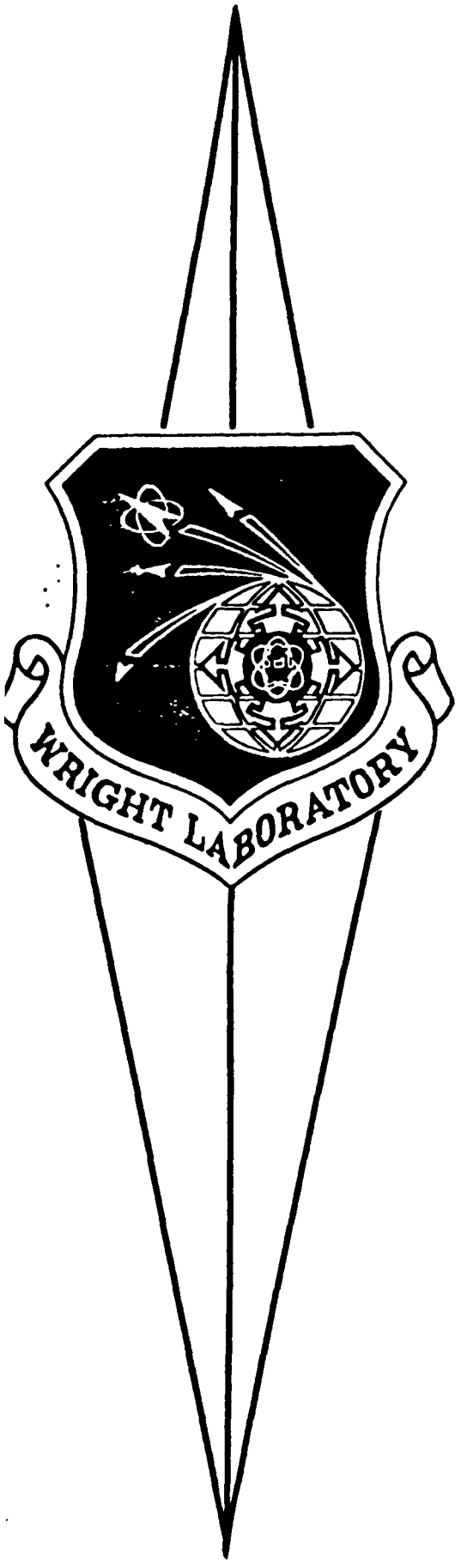
WL-TM-92-104

DECOMPOSED FUNCTION CARDINALITY OF SELECTED LOGISTIC FUNCTIONS



Timothy N. Taylor
WL/AART-2
WPAFB OH 45433-6543

System Concepts Group
Applications Branch
Mission Avionics Division

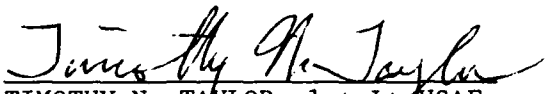
Avionics Directorate
Wright Laboratory
Wright-Patterson AFB OH 45433-6543

FOREWORD

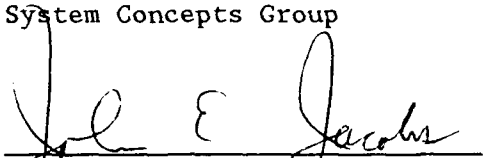
This technical memorandum was prepared by Timothy N. Taylor. This report documents an experiment in decomposing logistic functions (a set of functions which belong to the class of chaotic functions) and correlating their Decomposed Function Cardinality (DFC) with their Lyaponov exponent. This work was carried out in the System Concepts Group, Applications Branch, Mission Avionics Division, Avionics Directorate, Wright Laboratory, Wright-Patterson AFB, Ohio 45433-6543. This study was performed under work unit 0100AA13, Pattern Theory 2.

The author wishes to thank Dr Timothy Ross and Mr Michael Noviskey of WL/AART, Wright-Patterson AFB, Ohio. Additional thanks goes to Prof Jim Wolper and Ms Christine Yoon.

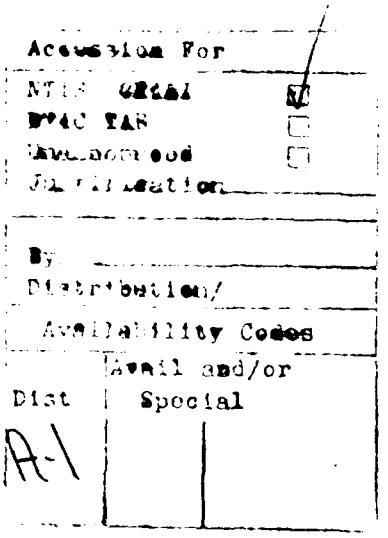
This Technical Memorandum has been reviewed and approved.



TIMOTHY N. TAYLOR, 1st Lt USAF
Electronics Engineer
System Concepts Group



JOHN E. JACOBS
Chief, System Concepts Group
Applications Branch



Decomposed Function Cardinality of Selected Logistic Functions

Timothy Taylor

April 7, 1992

1 Introduction

This memo documents the results of Pattern Theory 2 Task Order 3. The objective of this task was to decompose a set of logistic functions. In our prior experiments into the phenomonology of function decomposition (reported on in *Pattern Theory: An Engineering Paradigm For Algorithm Design* WL-TR-91-1060), we decomposed a wide variety of non-chaotic functions. The logistics functions decomposed in this task represent our first look at the ability of decomposed function cardinality (DFC) to measure complexity (or patternness) in a chaotic function. For each logistic function that we decomposed, we also calculated an approximation of the Lyaponov Exponent, a common measure of complexity in chaotic functions, and then computed the correlation between DFC and the Lyaponov Exponent over all functions.

2 Background

2.1 Introduction to Pattern Theory

Pattern Theory is primarily an approach to solving the basic problem of algorithm design: to begin with some abstract definition of a function (i.e., a formula, a table of values, a set of samples, etc.) and to end with a computer algorithm that realizes that function. It is our contention that what makes this problem solvable is the existence of some structure in the function that we call 'pattern-ness' and that if we can automate the process of discovering the underlying pattern in any patterned function, we can then automate the process of algorithm design.

There is a 'common sense' recognition that patterns occur in many forms: strings, sequences, images, etc. It seems that patterns are easier to remember, easier to extrapolate, and, also, easier to describe in a simplified form. This last concept relates pattern-ness to simplicity, or the inverse of complexity, and it is this concept we have chosen to focus on for two reasons. Firstly, computational complexity already has a well developed theory, and, secondly, it is quantifiable. There are many measures

of computational complexity. We feel that the one we've chosen, DFC, captures the essence of complexity in the sense of patterns. In the above-mentioned TR, the generality of DFC was supported both theoretically, by relating it to time complexity, program length, and circuit complexity, and experimentally by decomposing more than 1000 different functions of various types and different degrees of pattern-ness.

2.2 Function Decomposition

The decomposition of binary functions was first described by R. L. Ashenhurst in 1958. The first textbook on the subject was *A New Approach to The Design of Switching Circuits*, by H. L. Curtis published in 1962. By "function", we mean the traditional mathmatical function: a set of ordered pairs $(x, f(x))$ so that for every x there is exactly one $f(x)$. In general, functions may have more than one variable: $(f(x, y, z))$. The decomposition of a function is an expression of that function in terms of a composition of other functions. Suppose

$$f(x_1, x_2, \ldots, x_6) = F[\theta[\phi(x_1, x_2), \lambda(x_3, x_4)], \psi(x_5, x_6)]$$

then the RHS is the decomposition of f.

Decomposing binary functions was of particular use in reducing the number of circuit elements in switching circuit designs in the early 1960's. The method suggested by Curtis is useful for decomposing functions of five variables or less, but functions of six variables are difficult to decompose by hand, and functions of seven varibles or more are intractable. In the summer of 1989, Chris Vogt and Michael Noviskey designed and wrote a program in Ada which can decompose binary functions of up to ten variables. We refer to it as the Ada Function Decomposer (AFD) and it has ten versions which each approach the decomposition process with slightly different heuristics. On the AART VAX 11/780 it takes between half an hour and three hours to decompose a typical 8 variable function using the fastest version. A ten variable function might take several days. This program was used to produce the experimental results presented in this paper.

3 The Logistic Map

The Logistic Map is composed of a group of functions dependent on a constant μ which are of the form:

$$F_\mu(x_{i+1}) = \mu(x_i - x_i^2)$$

where μ is a real number less than 4.0. For any μ, an initial value for x can be chosen between zero and one. Each successive x is then dependent upon the preceeding x and is also bounded between zero and one.

For values of μ less than 3.0, the sequence will converge to a single point. When μ is slightly greater than 3.0, the sequence will converge to two points between which

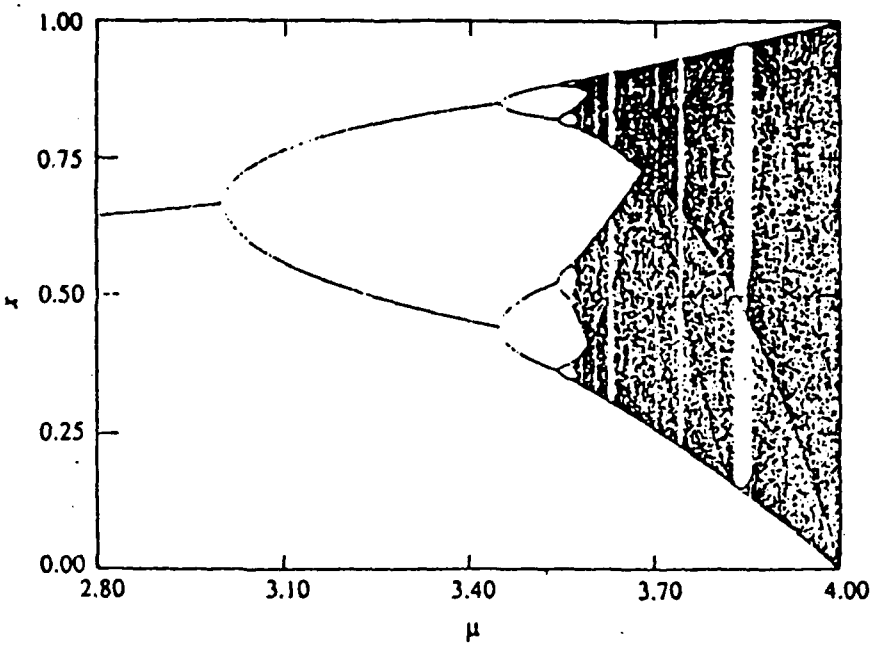


Figure 1: The Logistic Map

it will oscillate. For increasing values of μ up to approximately 3.5, the sequence converges to a fixed number of points (first four and then eight) between which it oscillates. For μ greater than approximately 3.55, there are regions in which the sequences are chaotic. Although the values in these chaotic sequences are bounded, the sequences never converge to any finite number of points. Interestingly, for a few discrete values of μ greater than 3.55, regions of stability again arise where the sequences will converge to a number of fixed points, but in these 'post-chaos' regions, the number of fixed points is always odd, rather than being a power of two.

A plot of the two-dimensional 'logistic map' as it is commonly shown is reproduced in figure 3. The sequences were produced by choosing an initial $x = .25$ and eliminating the first 200 transient terms.

4 The Experiment

There were two considerations in the design of the experiment. The first was, which particular functions from the logistic map should we choose to decompose? Because the decomposition process takes a significant amount of time, we had to be fairly selective in our choice of μ's. The second was, how best should we translate these sequences of real numbers into binary functions which we can decompose?

After some discussion, we initially chose roughly forty values for μ which fell into four general categories:

1. 3.5 to 3.9 at an interval of tenths and one point at 3.99

2. 3.61 to 3.85 at an interval of hundreths

3. 3.775 to 3.785 at an interval of thousandths

4. 3.825 to 3.835 at an interval of thousandths

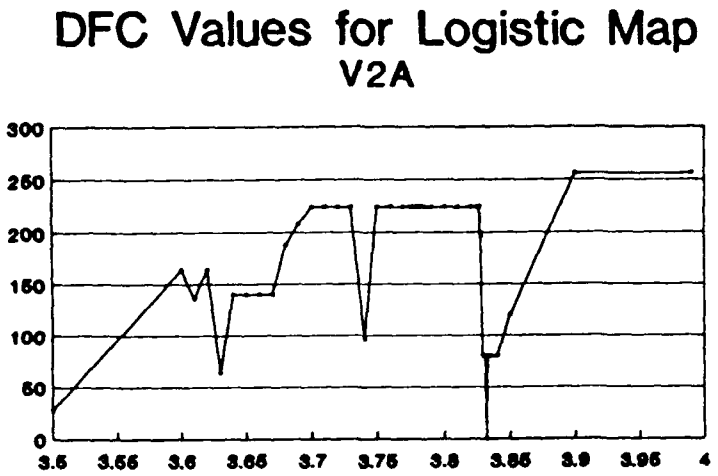
The first group was chosen to give a broad look at the generally increasing complexity in the logistic map. We were primarily concerned with the area of the logistic map which was chaotic, so values for μ less than 3.5 were not used. The second group was chosen to obtain a finer resolution in the chaotic region. At this point, enough detail is provided to see variations in complexity within the chaotic region. The last two groups were chosen specifically to compare the DFCs of the functions in each tight region to the Lyaponov exponents for functions there. It was known a priori (and it is obvious from looking at the logistic map) that in the first narrow region (3.775 to 3.785) the Lyaponov exponent is relatively high, indicating a good deal of complexity, and also that in the second region (3.825 to 3.835) there is a sudden dip in the Lyaponov exponent indicating a decrease in function complexity.

Each of these sequences was generated in the same way that the logistic map was generated: x_0 was chosen to be .25, and the first 200 transient terms were eliminated. A discussion of our confidence that the initial value chosen for x would change the binary function resulting from the sequence, but would not have an effect on the DFC of the function, is presented in Appendix B.

We answered the second consideration by using the same general method described in our *Pattern Theory* report when we decomposed other (non chaotic) sequences: The first step was to translate the sequences of real numbers into sequences of binary numbers. There are several ways to do this; in this case we applied a simple threshold: if the nth number in the real sequence was greater than .5 then the nth number in the binary sequence was chosen to be 1; if the nth number in the real sequence was less than or equal to .5, the nth number in the binary sequence was chosen to be 0. The second step then was to translate the binary sequence into a binary function. This was simply done by allowing each succeeding number in the sequence to be the succesive output of a function with a number of binary inputs equal to log_2 of the number of elements in the sequence. We chose to decompose sequences of 256 elements each. This translated the sequences into functions of eight binary variables where the nth element in the sequence was defined to be $f(x_1, \ldots, x_8)$ where $x_1^7 + \ldots + x_8^0 + 1 = n$.

Although some information is lost by doing this sort of transformation, a reasonable amount of information is still retained. For example, the binary sequence arising from the logistic function for $\mu = 3.5$ is a repetition of '0111', a cluster which captures the fourness of the double bifurcation at that point in the logistic map. At around $\mu = 3.83$ the binary sequence becomes a repetition of '011', a similarly simple cluster which captures the threeness of the logistic map at this point inside the region of chaos. In other regions where more complexity exists, there are no such simple analyses of the binary functions; however, there still exists a sense of the movement from the upper to the lower half of the logistic map.

Figure 2: DFC Values vs μ for Logistic Map



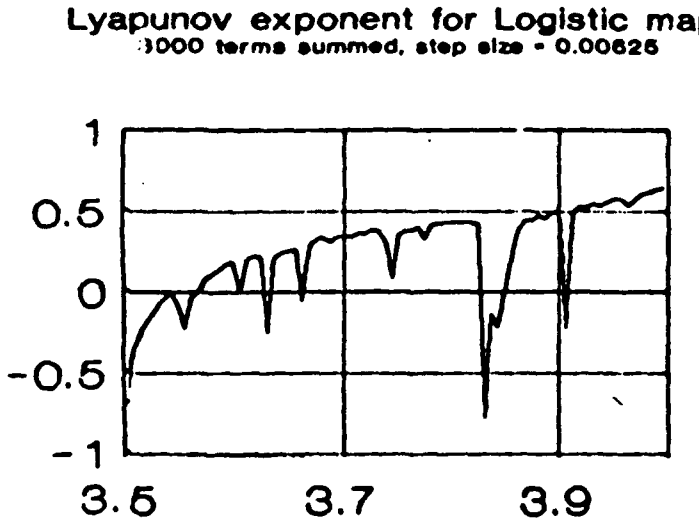
Once these decisions were made, a short program was written in Turbo Pascal to produce the binary functions corresponding to the chosen values of μ. The functions were then decomposed using the quickest AFD algorithm. In cases where the DFC seemed unusually high, a few functions were decomposed again, using a more exhaustive version.

4.1 The Results

The actual DFC values obtained for each region explored are shown graphically in the chart included in Figure ??. The first thing to notice is that decompositions were found for many of the sequences in the chaotic region, thus demonstrating the generality of DFC as a measure of complexity. It's visually apparent that the complexity of the logistic map increases as one moves toward the right, and if you follow the DFC values corresponding to $\mu = 3.5, 3.6 \ldots 3.99$ you will notice that they do show a steady increase. The intervals of hundreths between DFC values give us a fairly clear picture of the existence of regions of stability within the chaotic region. Although there is an overall steady increase in DFC with increasing μ, there are some values of μ (notably 3.63, 3.74 and 3.83) where the DFC suddenly drops. A quick glance at the logistic map will show that these are the same places where the functions become visibly less chaotic.

In *Chaotic Dynamics of Nonlinear Systems*, S. Neil Rasband suggests that the Lyaponov exponent is a good indicator of chaos. Our experiments have shown that there is a strong correlation between the DFC values and the Lyaponov exponent. The Lyaponov exponent for the logistic map is shown graphically in Figure ??. For the values of μ chosen in the third group of functions, the Lyapononv Exponent is

Figure 3: Lyaponov Exponent vs μ for Logistic Map



fairly stable and relatively high. The average DFC over the region was 224 (87.5 %DFC) and similarly stable. By contrast, for the values of μ chosen in the fourth group, the Lyaponov exponent is low with a sharp dip. The average DFC over the fourth region was only 118 (46.1 %DFC) and there is also a sharp dip in precisely the same place: $\mu = 3.832$.

In the course of the experiment, 49 different functions were produced and decomposed. After the decomposition was complete, we computed an approximation of the Lyapononv exponent for each value of μ and calculated the overall correlation. It was found to be .904. Further statistical analysis of the relationship between DFC and the Lyaponov exponent is shown in Figure ??.

The obvious and significant conclusion to be drawn from the experiment is that DFC *is* a robust measure of complexity for chaotic as well as non-chaotic functions.

μ	DFC	L.E.
3.5	28	-.8656
3.6	164	.1812
3.61	136	.1955
3.62	164	.1997
3.63	64	-.0136
3.64	140	.2284
3.65	140	.2482
3.66	140	.2841
3.67	140	.3070
3.68	188	.3456

Figure 4: DFC and L.E.

μ	DFC	L.E.
3.69	208	.3501
3.70	224	.3495
3.71	224	.3609
3.72	224	.3632
3.73	224	.3837
3.74	96	-.1041
3.75	224	.3566
3.76	224	.3830
3.77	224	.4033
3.775	224	.3306
3.776	224	.3760
3.777	224	.3922
3.778	224	.3863
3.779	224	.3845
3.78	224	.3985
3.781	224	.4082
3.782	224	.3961
3.783	224	.4132
3.784	224	.4152
3.785	224	.4170
3.79	224	.4262
3.80	224	.4359
3.81	224	.4245
3.82	224	.4325
3.825	224	.4242
3.826	224	.3996
3.827	224	.3510
3.828	196	.3129
3.829	80	-.1703
3.83	80	-.3682
3.831	80	-.6556
3.832	0	-1.325
3.833	80	-.6174
3.834	80	-.4228
3.835	80	-.3090
3.84	80	-.0447
3.85	120	.0082
3.9	256	.4867
3.99	256	.6368

Figure 5: DFC and L.E. continued. Correlation = .90432

Appendix B

It can be easily demonstrated that logistic functions which converge to a single point or a finite and small number of points will exhibit this convergence regardless of the initial value chosen for x. However, demonstrating that a chaotic logistic function retains the same degree of complexity, independent of the initial x, is a little more difficult. When the value of μ is sufficiently great that the function will not converge to any finite sequence, different initial x's will always generate completely different infinite sequences. We know that the Lyapunov exponent, which is a common measure of complexity in chaotic systems, is completely independent of the initial x, so we felt it would be good to check whether or not the DFC for any given μ was dependent on the initial x.

We chose $\mu = 3.8$ and generated four sequences, using initial x's of 0.25, 0.4, 0.6 and 0.8 and eliminating the first 200 elements. The functions were named 'log38', log38a', 'log38b' and 'log38c' respectively. The decompositions are shown on the following pages. Although the functions are clearly different and chaotic, it is also clear that they decompose to the same cost and in exactly the same way.

```
PECIFY FUNCTION TABLE VALUES:
----------------------------
nter 0 to input from a file, 1 from terminal,
r 2 to QUIT program: Name of function: How many input variables does the fu cti
nter decimal equivalent of binary input that has a true value: Enter decima eq
A  3
ate:
          7          31        1991
unction name:
og38a
umber of variables:
          8
          1          2          3          4          5          6
1101101111111111111101011011101101111011011111111110111101010101 1
1111011011101101111110111110101110101011101111111101101111111111111
1011110111011011110101111011010101011110101010110101010110111011101
1101101111011010111101101111011011110101111011011101110101010101

Cost =         256
--------------------------------------------------------------------------------
Better decomp found
          7          8         10         11         12          9
1111001010111101011111011101001
Cost =         32
          1          2          3          4          5          6          1(
0000001101000010010001101000000000000101101011000001010101000011

Cost =         64
          1          2          3          4          5          6          1:
0011100010110110101010111111011100000001111010100001101011100011

Cost =         64
          1          2          3          4          5          6          1:
0100110101000101011000010101000010010011110011100100010101100011

Cost =         64
Decomposition cost:         224
Node Information:
          0          1          2          3          4          5          (
          1          4          0          0          0          0          (
CPU Time:
    90.00
================================================================================
CPU Time:
    99.33
Node Information:
          0          1          2          3          4          5          (
          1          4          0          0          0          0          (
================================================================================
SPECIFY FUNCTION TABLE VALUES:
------------------------------
Enter 0 to input from a file, 1 from terminal,
or 2 to QUIT program: Thank you for using the PBML function Decomposer
```

```
PECIFY FUNCTION TABLE VALUES:
-----------------------------
nter 0 to input from a file, 1 from terminal,
r 2 to QUIT program: Name of function: How many input variables does the functi
nter decimal equivalent of binary input that has a true value: Enter decima. eq
A  3
ate:
          7          31        1991
unction name:
og38b
umber of variables:
          8
          1          2          3          4          5          6
1111101101011110101011011110101011011101110101111010110101110110
1010110111101101111101110111111111011101010101111111011101011011
1111011011110110111101011011110110111010111011011110101011010101
1101011111011011110110111011010111101110110111110101010111011011

Cost =          256
-------------------------------------------------------------------------------
Better decomp found
          7          8         10         11         12          9
1101110010110111111011011111100110
Cost =          32
          1          2          3          4          5          6          1(
0000110111111111111010110011000010101000101010110111010000010100 1

Cost =          64
          1          2          3          4          5          6          1:
0011001000010011001001100000110101010100000101001010000011101000 1

Cost =          64
          1          2          3          4          5          6          1:
0101011011100101011100001000111001010011101110101011111011110110

Cost =          64
Decomposition cost:          224
Node Information:
          0          1          2          3          4          5          (
          1          4          0          0          0          0          (
CPU Time:
    90.97
-------------------------------------------------------------------------------
CPU Time:
   100.41
Node Information:
          0          1          2          3          4          5          (
          1          4          0          0          0          0          (
-------------------------------------------------------------------------------
SPECIFY FUNCTION TABLE VALUES:
------------------------------
Enter 0 to input from a file, 1 from terminal,
or 2 to QUIT program: Thank you for using the PBML function Decomposer
```

```
 PECIFY FUNCTION TABLE VALUES:
 ----------------------------
:nter 0 to input f.om a file, 1 from terminal,
:r 2 to QUIT program: Name of function: How many input variables does the fu:cti
:nter decimal equivalent of binary input that has a true value: Enter decima: eq
:A  3
)ate:
          7          31        1991
'unction name:
.og38c
Number of variables:
          8
          1          2          3          4          5          6
1010110111011101010111011111101101111111101111101010110111101101
1111010110101101110110101111101110111101110101010111101010110111
1011010111011101010110111111010101111111111101011011110101111111
1111111011111110110101110111101010110101110111111011101011:011011

Cost =          256
-------------------------------------------------------------------------------
Better decomp found
          7          8         10         11         12          9
11011010011101111001111101111100
Cost =          32
          1          2          3          4          5          6          1(
00000001101100100000000110001011100001100011100001000100111111101

Cost =          64
          1          2          3          4          5          6          1:
00001010010100101100001000C10000010010101110001111100111100011000

Cost =          64
          1          2          3          4          5          6          1:
01110110110001011001101001101001001100001100101110110010100000010

Cost =          64
Decomposition cost:          224
Node Information:
          0          1          2          3          4          5          (
          1          4          0          0          0          0          (
CPU Time:
    88.71
-------------------------------------------------------------------------------
CPU Time:
    97.82
Node Information:
          0          1          2          3          4          5          (
          1          4          0          0          0          0          (
-------------------------------------------------------------------------------
SPECIFY FUNCTION TABLE VALUES:
------------------------------
Enter 0 to input from a file, 1 from terminal,
or 2 to QUIT program: Thank you for using the PBML function Decomposer
```